System Configuration Models: Towards a Specialization Approach

Maryam Mohammad-Amini, Thierry Coudert, Élise Vareilles, Michel Aldanondo

Available online at www.sciencedirect.com

ScienceDirect

IFAC PapersOnline CONFERENCE PAPER ARCHIVE

IFAC PapersOnLine 55-10 (2022) 1189–1194

System Configuration Models: Towards a Specialization Approach

M. Mohammad Amini*&. T. Coudert*. E. Vareilles**. M. Aldanondo*****

*INP-ENIT, University of Toulouse, Tarbes, France
(e-mail: maryam.mohammadamini@enit.fr, thierry.coudert@enit.fr)

**ISAE-SUPAERO, University of Toulouse, France

(e-mail: elise.vareilles@isae-supaero.fr)

***IMT Mines Albi, University of Toulouse, Albi, France

(e-mail: michel.aldanondo@mines-albi.fr)

Abstract: Nowadays, system configuration helps to achieve mass customization and manage a wide variety of systems. System configuration is based on a model that gathers all relevant knowledge for a family of systems. This knowledge model can be difficult to formalize and keep up to date; indeed, the knowledge must be made explicit, and can come from different departments and experts within an organization. In addition, it must reflect the different variants and options of a family of systems. Therefore, we try to answer the following question: how to better formalize and structure knowledge for system configuration to define configuration models and use the benefits of specialization in terms of modeling? Thus, in our proposal, we introduce the elements required to formalize the knowledge and define configuration models. This formalization will be done in a structured way using the association of ontology and Constraint Satisfaction Problem (CSP). We use the abilities of ontology to model knowledge of the different artifacts, their characteristics, and composition, and the abilities of CSP to model the relations between artifacts. In our proposal, we also define artifacts at different levels of abstraction using specialization which allows experts to detail or refine the formalized knowledge. We illustrate our proposals on a simplified but realistic example of a bicycle.

Keywords: System configuration, Knowledge formalization, Ontology, CSP, Configuration model, Specialization, Abstraction level

1. INTRODUCTION

Nowadays, system configuration helps to achieve mass customization and manage a wide variety of systems to meet the demands of the market. To do this, experts need to formalize the knowledge related to a family of systems, including the system architecture, components, subsystems, variants, options, and all the relationships between them. In the rest of the article, we refer to artifacts as systems, subsystems, or components (Guillon *et al.*, 2021). It is not easy for experts to formalize this knowledge and define configuration models that incorporate it. This formalization must be done in such a way as to make the formalized knowledge understandable, shareable, more easily updated, and usable in different models. Therefore, we try to answer the question of how to formalize and structure knowledge for system configuration to define configuration models and use the benefits of specialization in terms of modeling. In this way, we use the association of ontology and Constraint Satisfaction Problem (CSP). Ontologies are used to model, at different abstraction levels, the knowledge about artifacts, their characteristics, and how they are composed. We use CSPs to model the allowed relationships between different artifacts using constraints. In our proposal, we define the specialization of configuration models which, to the best of our knowledge, is not the subject of any work in the literature.

The remainder of this paper is structured as follows. In section 2, we present the related works. In section 3,we propose the elements needed to model knowledge for system configuration. In section 4, we formalize knowledge for bicycle example. In section 5, we present the conclusion and future perspectives.

2. RELATED WORKS

In this section, the relevant articles on system configuration, knowledge formalization, approaches such as ontology and CSP, their association, and the specialization/generalization relationship are discussed.Then the scientific gaps are found.

2.1 System configuration

(Soininen *et al.*, 1998) mentioned product configuration as a task “can be roughly defined as the problem of designing a product using a set of predefined components while considering a set of restrictions on how the components can be combined.” Regarding this definition and making it more general, the set of artifacts is bounded and their connection is only allowed in specific ways. One of the pillars of

10.1016/j.ifacol.2022.09.551

configuration problems is the configuration model. (Oddsson and Ladeby, 2014) defined a product configuration model as "an abstract representation, describing the structure of the product, the entities the product consists of, and the rules on how the entities and their properties can be combined." In our work, all the definitions in product configuration are valid for system configuration. In configuration problems, two steps of knowledge formalization and knowledge reuse are considered. In this paper, we only focus on knowledge formalization.

2.2 Knowledge formalization

Knowledge formalization means that knowledge is structured and formalized in a way that facilitates its interpretation. In knowledge formalization (Fig. 1), experts are responsible to model knowledge and define configuration models. Configuration models must be correct and consistent representations of the system knowledge, i.e. system architecture, relationships between systems characteristics, the list of all options and alternatives... (Sabinand Weigel, 1998) mentioned that formalizing different types of relationships between artifacts such as classification (is-a), aggregation (part-of), and relationships related to cardinality, geometry, and so on is difficult. Since the formalization of knowledge to define configuration models is a critical issue, many efforts have been made to formalize it. For example, (Hotz *et al.*, 2014) presented various knowledge representation approaches such as Constraint-Based (CSP, Dynamic Constraint Satisfaction, Generative Constraint Satisfaction), Graphical (Feature Models and Unified Modeling Language – UML models), and Logic-Based (First Order Logic, Answer Set Programming) to define a configuration model. (Soininen *et al.*, 1998) proposed a general ontology containing modeling concepts to represent knowledge in the domain of configuration. (Cao and Hall, 2020) proposed an ontology-based method to configure modular buildings. (Yang, Dong and Miao, 2008) proposed an ontology-based approach to formalize the knowledge of product configuration using Web Ontology Language (OWL) and Semantic Web Rule Language (SWRL). (Esheiba *et al.*, 2021) proposed a recommender system using ontologies to capture production-related knowledge and using CSP to encode product-service systems variants. (Bischof *et al.*, 2018) used Shapes Constraint Language to model constraints but updating the formalized knowledge is not easy. Based on the literature, CSP is the most widely used approach to formalize knowledge in configuration problems. Ontology is another famous approach to model knowledge in this domain.

Ontology Definition: So far, different definitions have been mentioned for ontology regarding the various contexts. (Studer, Benjamins and Fensel, 1998) defined an ontology as a "formal, explicit specification of a shared conceptualization." An ontology captures domain knowledge to provide a common understanding of it. It defines concepts or classes, attributes of classes, attributes domains, hierarchical relationships, abstraction levels and inheritance, compositions, rules, axioms, and assertions. In our work, although ontologies can be used to define artifacts at different abstraction levels and their relationships, they cannot represent all the constraints needed to formalize knowledge for system configuration e.g., compatibility between artifacts or artifacts characteristics.

CSP Definition: (Montanari, 1974) defined a CSP as a triplet

{X, D, C} where X is a set of variables, D is a set of domains of variables – one for each variable, and C is a set of constraints. Constraints represent restrictions on the combination of variable values. (Felfernig et al., 2014) mentioned that CSP provides several advantages, including distinguishing between the modeling part and the solving part, and defining various variables and constraints with different types. In our work, while CSPs can be useful for modeling all the constraints required to formalize knowledge for system configuration, i.e., incompatibility between artifacts or artifact characteristics, they cannot represent hierarchical relationships, abstraction levels, and inheritance.

Association of Ontology and CSP: Based on the literature, the lack of paper on the knowledge formalization for system configuration using the association of ontology and CSP is obvious. In this work, we use the association of ontology and CSP to formalize knowledge which enables experts to model knowledge in a structured manner. Ontology can be used to model knowledge about different artifacts i.e. system, sub-system, components, specialization of artifacts, characteristics, and also their relationships (such as composition) (Fig. 1, highlighted in purple). CSP can be used to model different types of relationships between different artifacts and/or artifacts characteristics using constraints (Fig. 1, red lines).

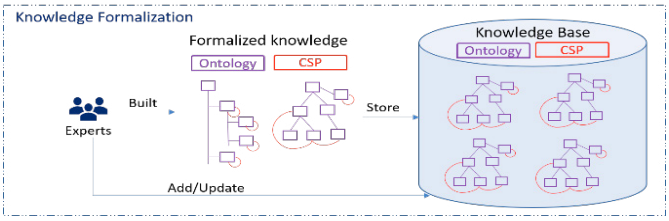

Figure 1. Knowledge formalization

2.3 Specialization/Generalization relationship

(Ohira, Hochin and Nomiya, 2011) mentioned specialization enables the creation of new artifacts downwards that is explicitly derived from an existing general artifact. Artifacts that are far away from the general artifact are more specialized. Each created artifact in addition to the characteristics inherited from the general artifact may have specific characteristics only allocated to itself. (Ohira, Hochin and Nomiya, 2011) represented that generalization is the process of reversing specialization. In this paper, we focus on specialization. Although specialization of artifacts is mentioned in some articles, e.g. on ontology or UML, to the best of our knowledge there is no paper about the specialization of models into more specific models in the context of configuration.

Related works Synthesis: In this section, after reviewing the relevant papers, we found that there is a lack of work on 1) knowledge formalization for system configuration using the association of ontology and CSP, and 2) specialization of configuration models.

3. PROPOSALS

In this section, the different elements used to model knowledge for systems configuration are proposed. Two kinds of elements are formalized: Generic Artifacts (GA) and Generic Models (GM). The specialization of both elements is presented.

3.1 Generic Artifact

GA Definition: In the proposed model, as mentioned in the introduction, artifacts are systems, sub-systems, or components. Sub-systems can be decomposed into sub-systems and/or components while a component is not decomposed. They correspond to tangible parts of systems to be configured (Guillon et al., 2021). A Generic Artifact or GA is a family of artifacts with common characteristics (Ohira, Hochin and Nomiya, 2011). A GA is described using attributes for which the possible values are defined by their validity domains. Attributes can be symbolic or numerical and their domains can be discrete or continuous. It exists specific attributes corresponding to the Key Performance Indicator (KPI) such as the cost, the weight, and the performance, to assess GAs. Some relations between attributes allow or forbid some combinations of attribute values and then define what the possible characteristics of the artifacts of the GA family are.

We propose that within a GA, knowledge is formalized using CSP. As mentioned before, CSP is defined by a triplet {variables, domains, constraints} which can easily be matched with attributes, domains, and relations of GAs. The use of CSP within a GA, allows us to use filtering mechanisms (Bessière, 1994), to restrict attribute domains by keeping only consistent values. We use the terms attributes, domains, and constraints in the rest of the paper.

GA Specialization: In the proposed model, GA can be defined at different levels of abstraction using specialization relationships. A GA or GA_{Child} can be specialized from a more general GA or GA_{Parent} to refine or detail GA_{Child} knowledge. In such a case, first, GA_{Child} inherits all the characteristics of GA_{Parent}: the attributes, their domains, and the constraints. Second, the valid domains of GA_{Child} can be specialized (or narrowed) by (1) restricting its inherited domains thanks to inherited constraints modifications (addition of forbidden combinations of values) or (2) by defining new constraints on its attributes. In the case where the specialized GA_{Child} has no specific constraints, its valid domains are equal to the domains of its parent GA_{Parent}. Third, in addition to what a specialized GA_{Child} inherits from its parent GA_{Parent}, specific attributes, domains, and constraints that are only dedicated to that specialized GA_{Child} and which are modeling its specific knowledge can be added. Within a specialized GA_{Child}, inherited knowledge and specific knowledge can be combined thanks to constraints linking inherited attributes and specific attributes.

The proposed model allows for defining general GA_{Parent} which gathers all the common characteristics of more specific $GA_{Children}$. The modification of characteristics of any GA_{Parent} (attributes, domains, constraints) is inherited by all its specialized $GA_{Children}$ and their descendants.

3.2 Generic Model

GM Definition: A Generic Model or GM allows to formalize knowledge about a family of systems that have common characteristics, including all possible options and alternatives. Following (Oddsson and Ladeby, 2014), a GM represents:

1. the generic architecture, i.e. the generic Bill of Materials (BOM), of the family of systems. A BOM is a list of artifacts (components and subsystems) with the quantities of each needed to manufacture a specific system. Following the BOM hierarchical nature, a GM formalizes the decomposition links connecting a system GA (the root of the BOM and at level 0), corresponding to a generic system, to all the GA composing it (sub-systems or components), on k decomposition levels (from *1* to *k*). This decomposition is made using the type of association "is composed of." In the following, we refer to GM or system GA for the highest-level generic system (level *0*) of the BOM.

2. the relations between all the GA composing the GM, regardless of their level of decomposition. These relations can be defined directly between several GAs, such as "mandatory", "forbidden", "excluded" or between GA's attributes, such as allowed or forbidden combinations of attribute values between two or more GAs. Relations between GAs allow formalizing all the system family solution space, i.e. all the systems that can be manufactured regarding the GM knowledge.

3. the method for assessing the GM's KPIs. As each GA is assessed on KPIs, the way to aggregate them from one level of decomposition *n+1* to the direct upper-level *n* must be defined. It can be a simple sum or a much more complicated formula (MAX, MIN, AVERAGE, etc.).

Within a GM and for GA, knowledge is formalized as a CSP to model i) the constraints between GAs; ii) the constraints between the values of the attributes of different GAs composing the GM; iii) the constraints linking the KPIs. The use of CSP within a GM allows local filtering on GAs, via constraints embedded in GA, and propagations over the entire GM, via constraints between GAs or attribute values of multiple GAs. The CSPs thus guarantee the overall consistency of the GM (or system GA) concerning the different knowledge of its GAs.

KPI assessment: We have seen that GM and GA are assessed on KPIs (e.g. weight, performance, price, cost, etc.). The way to aggregate them from one level *n+1* to the direct upper-level n has to be defined at the GM level, regardless of the number of GAs composing the GM. Such a relation is therefore formalized as a global constraint (Rossi *et al.*, 2006) to avoid an exhaustive inventory of all the GAs involved. This global constraint is applied at each level of decomposition for each GA. Equation (1) illustrates the computation of a KPI named $GA.KPI_j$ for a GA composed of several GA_i. The SUM operator is used in (1).

$$GA.KPI_j = \left(GA_i.KPI_j * quantity\ of\ GA_i\right) \quad (1)$$

GM Specialization: To efficiently model knowledge for system configuration, as for GA, GM can be defined with

different levels of abstraction. A GM or GM_{Parent} can be specialized into one or more specific GMs or $GM_{Children}$ to refine or detail GM_{Parent} knowledge. That will allow us to define more specific models corresponding to more specific systems and their configurations. The benefits of GM specialization are first the automatic update of knowledge from one GM_{Parent} to all its specialized $GM_{Children}$. Second, specialization avoids knowledge experts to start the blank page to formalize the knowledge. Several abstraction levels of knowledge on the same family of systems can thus be formulated and reused.

Firstly, while specialization, a specialized GM_{Child} inherits all the characteristics of its parent GM_{Parent}, i.e. the architecture of GM_{Parent} (BOM), all the GAs composing it with their embedded CSP, all constraints linking GAs or GAs attributes, and all the KPI aggregation methods. As for GA, inherited constraints between GA or GA attributes can be updated by the modification of tuples to restrict the allowed combinations, or new specific constraints between inherited GAs or GA attributes can be defined.

Secondly, an inherited GA can be specialized within a GM_{Child} compared to GM_{Parent}. To detail knowledge, it is sometimes necessary to specialize a GA_{Parent} to one of its children GA_{Child} (if they exist). The knowledge of GA_{Child} will be more precise and accurate than the one of its parent GA_{Parent}. This specialization mechanism is only allowed for GAs belonging to the same family line: the more specialized GA_{Child} must be a descendant of the inherited GA_{Parent}. Indeed, the set of inherited attributes and constraints must remain the same between GA_{Child} and GA_{Parent} for the overall consistency of the GM_{Child}.

Thirdly, one or more specific GAs which are only dedicated to that specialized GM_{Child} and which are modeling its specific knowledge can be included in the GM_{Child} architecture, at any level of decomposition. Inherited GA and specific GA can be combined thanks to new constraints linking their attributes. KPIs of the direct upper level will automatically consider these new GA in their assessments and allow the evaluation of the overall GM_{Child}.

As mentioned for GA, within a GM, filtering mechanisms ensure the overall consistency of GAs, regardless of their specialization or decomposition level, and finally of the GM, whatever the level of specialization. The proposed model combined with inheritance mechanisms allows us, in a very simple but efficient way, to define GM with several specialization levels. GMs are based on GAs by architecting them and linking them to form a consistent generic model of a family of systems. Each GM can be specialized by (1) adding attributes and/or constraints, (2) specializing some GAs (in the same family line and descendants only), (3) adding new GAs at any level of the BOM and any level of abstraction, and (4) adding constraints between inherited and added GAs. The modification of characteristics of any GM_{Parent} (architecture, GA, constraints) is inherited by all its specialized $GM_{Children}$ and their descendants. This allows experts to design and update GM in a very easy way.

4. CASE STUDY

To illustrate our proposals, knowledge formalization for the configuration of bicycles is presented. It is considered that a bicycle is only composed of one saddle, one frame, two wheels, and one or two brakes. First, we define the relevant GAs with their attributes, domains, and constraints. Second, we show the specialization of *Wheel GA* into *City Wheel GA* and *Mountain Wheel GA*. Third, we define a *Bicycle GM* and we illustrate its specialization into *City Bike GM*.

Illustration of GA Definition: To define knowledge in the domain of bicycles, we need to define all the important GA such as *System GA*, *Bike GA*, *Frame GA*, *Saddle GA*, *Wheel GA*, *Brake GA*, *Rim GA*, and *Tire GA*, as shown in Fig. 2. Our example illustrates GA and GM definition, as well as GA and GM specialization, on simple examples. We, therefore, define only the *System GA*, *Rim GA*, *Tire GA*, and *Wheel GA*, and their essential characteristics as follows:

The *System GA* is defined as the most general GA and can be specialized into other GAs. In our case, it has only one attribute which is the KPI "Cost", notated *System.Cost*, with the validity domain {[0, 15000]}. As all the other GAs are a specialization of the System GA, they inherit the KPI Cost with the same domain.

The *Rim GA* is only characterized by two attributes: *Rim.Size* with domain {[12, 29]}, and *Rim.Cost* with domain {[0, 15000]} which is inherited from System GA. The cost of the *Rim GA* depends on its size. A constraint representing the compatible values of *Rim.Size* and *Rim.Cost* is made explicit as shown in Table 1.

Table 1. Constraint of *Rim GA*

Rim.Size	*Rim.Cost*
{18, 19}	[290, 320]
{13, 14, 20, 23, 26}	[270, 650]
{16, 17}	[650, 800]

Filtering this constraint leads to restricting the initial domains of the attributes *Rim.Size* and *Rim.Cost*, respectively to {[13, 14], [16, 20], 23, 26} for *Rim.Size* and {[270, 800]} for *Rim.Cost*.

The *Tire GA* is only characterized by two attributes: *Tire.Size* with domain {[12, 29]} and *Tire.Cost* with domain {[0, 15000]} which is inherited from *System GA*. The cost of the *Tire GA* depends on its size. A constraint representing the compatible values of *Tire.Size* and *Tire.Cost* is made explicit as shown in Table 2.

Table 2. Constraint of *Tire GA*

Tire.Size	*Tire.Cost*
[12, 17]	[10, 20]
[18, 29]	[18, 40]

Filtering this constraint leads to restricting the initial domains of the attribute *Tire.Cost* to {[10, 40]}.

The *Wheel GA* is characterized by three attributes: *Wheel.Diameter* with domain {[12, 29]}, *Wheel.Material* with domain {Aluminium alloy, Steel, Carbon fiber}, *Wheel.Cost* with domain {[0, 15000]} which is inherited from *System GA*. Since any wheel diameter cannot be associated with any material, we need to formalize this relationship by a constraint that links the *Wheel.Diameter* to its *Wheel.Material* (Table 3). We also associate the *Wheel.Cost* to that constraint.

Table 3. Constraint of *Wheel GA*

Wheel.Diameter	*Wheel.Material*	*Wheel.Cost*
{12, 14, 16, 18, 20, 24}	{Aluminum alloy}	[270, 3000]
{26, 27.5, 29}	{Aluminum alloy, Steel, Carbon fiber}	[290, 4000]

Filtering this constraint leads to restricting the initial domains of the attributes *Wheel.Diameter* to {12, 14, 16, 18, 20, 24, 26, 27.5, 29} and *Wheel.Cost* to {[270, 4000]}.

Illustration of GA Specialization: To detail *Wheel GA* knowledge, we specialize it into *CityWheel GA,* as shown in Fig. 2. As explained in section 3.1, *CityWheel GA* inherits from its parent *Wheel GA* all its characteristics: attributes, domains, and constraints. *CityWheel GA* is now characterized by three inherited attributes with filtered domains: *CityWheel.Diameter* with domain: {12, 14, 18, 16, 20, 24, 26, 27.5, 29}, *CityWheel.Material* with domain: {Aluminium alloy, Steel, Carbon fiber}, *CityWheel.Cost* with domain: {[270, 4000]} and an inherited constraint that links the City*Wheel.Diameter* *CityWheel.Material* and *CityWheel.Cost* (Table 3). The constraint presented in Table 3 can be more specialized by the addition of tuples, as shown in Table 4 specialized tuples are explicitly shown with a gray background.

Table 4. Inherited constraint of *CityWheel GA*

CityWheel.Diameter	*CityWheel.Material*	*CityWheel.Cost*
{12, 14, 16, 18, 20, 24}	{Aluminum alloy}	[270, 3000]
{26, 27.5, 29}	{Aluminum alloy, Steel, Carbon fiber}	[290, 4000]
29	Carbon fiber	[3500, 3700]
29	Steel	[1000, 2500]

To illustrate the addition of new attributes and constraints in a GA_{Child}, we consider that *CityWheel GA* has optional reflectors. This option is modeled by the addition of a new attribute *CityWheel.Reflector* with domain {[0, 6]} (corresponding to the number of reflectors) and a new constraint linking the city wheel diameter and the number of reflectors, as shown in Table 5.

Table 5. New constraint of *CityWheel GA*

CityWheel. Diameter	*CityWheel.Reflector*
[12, 17]	{[0, 4]}
[18, 29]	{0, [3, 6]}

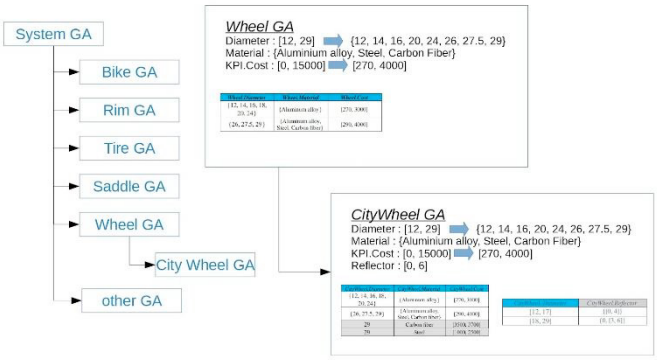

Figure 2. Specialization of *Wheel GA*

Illustration of GM Definition: Bike GM represents a family of Bikes using its architecture of GAs, their quantity, the constraints between GAs, and the aggregation methods for KPIs. As shown in Fig. 3, Bike GM is composed of 1 Saddle GA, 1 Frame GA, 1 or 2 Brake GA, and 2 Wheel GA. A Wheel GA is composed of 1 Rim GA and 1 Tire GA.

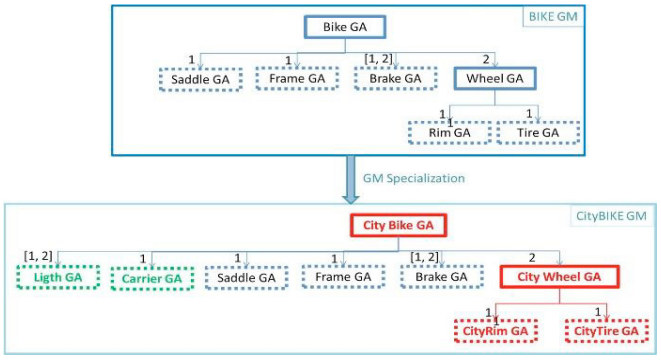

Figure 3. Specialization of *Bike GM*

All the GAs can be linked by constraints to model the system family solution space, i.e. all the systems that can be manufactured regarding the GM knowledge. For the illustration, we have defined a constraint between three attributes of three GAs: regarding the fact that for a *Wheel GA*, the *Tire GA* is mounted on the *Rim GA*, they must have the same diameter otherwise *Tire GA* mounting is not possible. This knowledge is defined by (2) which represents the fact that the diameters of *Wheel GA*, *Tire GA*, and *Rim GA* must be equal. This constraint is embedded in the *Wheel GA* which is decomposed.

$$Wheel.Diameter = Tire.Diameter = Rim.Diameter \quad (2)$$

The filtering of constraint (3) has an impact on the domain of the three GAs. *Wheel.Diameter*, *Tire.Diameter* and *Rim.Diameter* are now equal to {14, 16, 18, 20, 26}. This reduction has no impact on the rest of the model.

The aggregation of the KPI cost is made from one GA_{Parent} thanks to the sum of all the KPI Cost of its GA_{Child}: GA_{Parent}.Cost = (GA_{Child}.Cost * Qty of GA_{Child})

This aggregation method applied to the *CityWheel* implies that the *CityWheel.Cost* = *CityRim.Cost* + *CityTire.Cost*. The

filtering approach deduces that → = [270, 800] * 1 ⊕ [10, 40] * 1 = [280, 840] instead of [270, 4000].

The *Bike GM* is now defined and can be specialized in a *CityBike GM* for instance.

Illustration of GM Specialization: To detail *Bike GM* and create a *CityBike GM*, we specialize *Wheel GA*, *Rim GA,* and *Tire GA* respectively into *CityWheel GA*, *CityRim GA,* and *CityTire GA*. All the characteristics (attributes, domains, and constraints) of these more specialized GAs are now considered in the *CityBike GM*, as shown in Fig. 3. The *CityWheel.Reflector* and all the specialized constraints are now part of the *CityBike* model. This specialization of some GAs into *CityBike GM* has no impact on the attribute's domains. Specific GAs can be included in a GM_{Child} architecture. In the case of the *CityBike GM*, we add two specific GAs: one corresponding to *Carrier GA* with a quantity of 1, and one corresponding to *Light GA* with a quantity between 1 and 2. As these new GAs have KPI.Cost, they are considered automatically in the assessment of the *CityBike GM*. In our example, there is no impact on the domains of the attributes.

Case Study Synthesis: In this section, first, GA definition and GA specialization are shown on a wheel example. Then GM definition and GM specialization are also presented on a bike example. The constraint filtering, an integrated activity of our approach, is run to keep only consistent values for a GM.

6. CONCLUSIONS

In this article, we have studied the formalization of knowledge in system configuration. As far as we know, no scientific work formalized the specialization of configuration models. This article aimed to propose the necessary elements to formalize knowledge and define system configuration models, and then apply specialization of models to refine knowledge and make models more specific. For this purpose, first, we defined the Generic Artifact (GA) to model knowledge of a family of artifacts. Second, we defined the specialization of GAs to refine or detail knowledge. Third, using GAs, we defined the Generic Model (GM) to formalize the knowledge of a family of systems with all possible options and alternatives. We defined the knowledge within the GA, and the GM using CSP, and used constraint filtering to maintain consistent values. The contributions of this paper include (1) formalizing knowledge for system configuration using the association of ontology and CSP (2) defining GMs at different levels of abstraction using specialization (3) using constraint filtering to keep GAs and GMs consistent. As future perspectives, we intend to work on (1) the development of our proposal to consider generalization, (2) the coding of our proposal in a real mock-up, and (3) the consideration of Engineer-to-Order configuration (ETO). The implementation of our proposal on real industrial cases is also in our minds.

REFERENCES

Bessière, C. (1994). Arc-consistency and arc-consistency again. in *Artificial Intelligence*, Volume (65), Issue 1.

Bischof, S., Schenner, G., Steyskal, S., and Taupe, R. (2018). Integrating semantic web technologies and ASP for product configuration. in *CEUR Workshop Proceedings*, pp. 53–60.

Cao, J., and Hall, D. (2020). Ontology-based product configuration for modular buildings. in *ISARC. Proceedings of the International Symposium on Automation and Robotics in Construction*, pp. 171–176.

Esheiba, L., Elgammal, A., Helal, I., and El-Sharkawi, M. E. (2021). A hybrid knowledge-based recommender for product-service systems mass customization. *Information*, 12(8), p. 296.

Felfernig, A., Hotz, L., Bagley, C., and Tiihonen, J. (2014). *Knowledge-based configuration from research to business cases*, Newnes.

Guillon, D., Ayachi, R., Vareilles, É., Aldanondo, M., Villeneuve, É., and Merlo, C. (2021). Productץservice system configuration: a generic knowledge-based model for commercial offers. *International Journal of Production Research*, 59(4), pp. 1021–1040.

Hotz, L., Felfernig, A., Stumptner, M., Ryabokon, A., Bagley, C., and Wolter, K. (2014). Configuration knowledge representation and reasoning. in *Knowledge-Based Configuration*, Elsevier, pp. 41–72.

Montanari, U. (1974). Networks of constraints: fundamental properties and application to picture processing. *Information Sciences*, 7, pp. 95–132.

Oddsson, G. and Ladeby, K. R. (2014). From a literature review of product configuration definitions to a reference framework. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing: AIEDAM*, 28(4), pp. 413–428.

Ohira, Y., Hochin, T. and Nomiya, H. (2011). Introducing specialization and generalization to a graph-based data model. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 6884 LNAI(PART 4), pp. 1–13.

Sabin, D. and Weigel, R. (1998). Product configuration frameworks - A survey. *IEEE Intelligent Systems and Their Applications*, 13(4), pp. 42–49.

Soininen, T., Tiihonen, J., Männistö, T., and Sulonen, R. (1998). Towards a general ontology of configuration. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing: AIEDAM*, 12(4), pp. 357–372.

Studer, R., Benjamins, V. R. and Fensel, D. (1998). Knowledge engineering: principles and methods. *Data and Knowledge Engineering*, 25(1–2), pp. 161–197.

Yang, D., Dong, M. and Miao, R. (2008). Development of a product configuration system with an ontology-based approach. *CAD Computer Aided Design*, 40(8), pp. 863–878.